VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Bakalářská práce

2012 **Jiří Strakoš**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2012 **Jiří Strakoš**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Strakoš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: CATHEDRAL Software, s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. RNDr. Petr Šaloun, Ph.D.**

Konzultant bakalářské práce: Ing. Josef Šustr

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

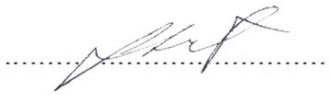
doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 17.4.2012



Jiří Strakoš

Abstrakt

Práce se zabývá samostatně vypracovanou praxí ve firmě Cathedral Software, kde jsem působil jako vývojář testovacích skriptů pro firemní software ArisCAT.

Jako vývojář testových skriptů jsem měl k použití program zakoupený firmou pro možnost automatického testování pomocí skriptů s názvem TestComplete verze 7. V práci je popsáno seznámení a práce s těmito softwary. Jejich funkce, možnosti a použití. Nakonec mnou vypracované závěrečné testovací skripty budou dále firmě sloužit v dalším vývoji jejich softwaru.

A v poslední řadě také zaškolení firemních programátorů pro jejich následnou samostatnou práci na testování.

Klíčová slova

Testování, testovací skripty, C# skript, Delphi, automatizované testování, ArisCAT, TestComplete, tester

Abstract

The thesis discuss of individual professional practice in the company Cathedral Software, where i work as developer of testing scripts for company software ArisCAT.

As developer of testing scripts I used program with automatic scripting capability named TestComplete version 7, which has been bought by company earlier. In thesis is described familiarization and work with these programs. Their functions, capabilities and usage. Finally my own produced scripts for testing will be used by company for future testing and development of their software.

After all also train company programmers for their subsequent individual testing.

Keywords

Testing, testing scripts, C# script, Delphi, automated testing, ArisCAT, TestComplete, tester

Seznam často používaných zkratek a pojmů

GUI (Graphical User Interface) – je uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků.

IDE (Integrated Development Environment) – vývojové prostředí většinou zaměřené na jeden konkrétní programovací jazyk, obsahující editor zdrojového kódu, kompilátor a většinou také debugger.

IntelliSense – je implementace automatického dokončování slov programovacího jazyka, které slouží jako dokumentace a rozlišování proměnných, funkce a metody užívajíce reflexi.

CVS (Concurrent Version System) - systém, který slouží ke správě verzí projektu.

IS (Informační Systém) – je soubor technologických prostředků a metod, které zabezpečují sběr, přenos, zpracování a uchování dat za účelem tvorby prezentace informací pro potřeby uživatelů.

Log (neboli Žurnál) - je název pro záznam nebo soubor záznamů, které si některé programy vytvářejí pro ukládání informací o své činnosti a běhu. Logy slouží při zpětné analýze k rozpoznání, zda došlo k nějaké chybě a pakliže ano, pak pomáhají určit, k jaké chybě došlo a proč.

SQL (Structured Query Language, česky strukturovaný dotazovací jazyk) – je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.

Seznam obrázků

Obsah

1. Úvod

Firma Cathedral Software se zabývá vývojem a implementací informačních systémů a to především o vývoj a dodávku modulárního informačního systému pro obchodní a výrobní firmy ArisCAT. Při praxi ve firmě Cathedral software na pozici vývojáře testovacích skriptů jsem dostal za úkol vytvořit automatizované testovací skripty, kterými by se měla ověřit implementace a funkčnost softwaru ArisCAT.

Prvním úkolem bylo seznámit se s programem ArisCAT. Musel jsem nastudovat dokumentaci a také si odzkoušet jeho funkčnost. Vše jsem konzultoval s programátory, ať nedojde k nějakým nedorozuměním. Dále jsem nastudoval z manuálu a internetu možnosti a použití testovacího programu TestComplete. Následující prací bylo samotné vytváření testovacích skriptů. Následné testování jejich funkčnosti a kontrola výstupních dat z automatizovaných testů. Výsledné skripty jsem konzultoval se zaměstnavatelem a programátory. Vysvětlil jim základní funkčnost programu TestComplete. Jak můžou vytvářet své skripty a dále s nimi pracovat. A debatovali jsme nad nasazením do modelu vytváření softwaru ArisCAT.

Výsledkem bakalářské práce je osvojit si techniky automatizovaného testování, vyzkoušet si metody a techniky používané při testování aplikací a následně spolupráce s programátory a začlenění tohoto testovacího cyklu do praxe.

2. O společnosti

Společnost Cathedral Sofware vznikla v polovině roku 1996 jako sdružení fyzických osob Cathedral Software & Hardware a její sídlo bylo vždy v Prostějově. Firma disponuje ryze českým kapitálem a její orientace byla od vzniku zaměřena na dlouhodobou spolupráci spojenou s tvorbou kvalitního softwaru na zakázku a poskytování ostatních služeb dle požadavků zákazníků.

V souvislosti s požadavky zákazníků firma rozšířila portfolio i o dodávku počítačových komponent a sestav. V letech 1997-1998 začala s dodávkami, instalacemi a správou informačních systémů. Zabývají se návrhem, realizací a výstavbou počítačových sítí pomocí strukturované kabeláže a realizací virtuálních privátních sítí. V roce 1999 se firma přestěhovala do nových prostor v zrekonstruovaném obchodním a společenském centru Atrium v Prostějově. Přesídlení jí přineslo nové programátory a grafiky. Díky tomu začala poskytovat služby spojené s webdesignem.

Pracovníci firmy Cathedral Software se ale nejintenzivněji zaměřovali vždy na vývoj informačních systémů tvořených dle přání a potřeb právě zákazníků. Jako jeden z prvních takových byl informační systém Žaluzie, který byl určen pro výrobce žaluzií. Dále pak Hokej 2000, který byl vytvořen s podporou Svazu ledního hokeje ČR.

Dále firma značně spolupracuje se společností LCS International a.s., která patří mezi největší producenty podnikových aplikací na českém trhu. Cílem této spolupráce je vývoj a následná distribuce integrovaného modulu pro komunikaci s informačním systémem Helios, jenž bude zajišťovat zpracování obchodů s těžce konfigurovatelnými výrobky, jako jsou například žaluzie, rolety, vrata a plastová okna.

Zlomový informační systém ale byl bezesporu IS ArisCAT, kterým se firma zabývá už od roku 2001. Tento systém je firmou vyvíjen dodnes a především je optimalizován na zpracování již dříve zmíněných obtížně konfigurovatelných výrobků. S tímto systémem jsem pracoval a prováděl na něm testování dle zadání BP. Detailnějším informacím a funkčnosti tohoto systému bude věnována jedna z dalších kapitol.

3. Automatizované testování

Rozšíření softwaru v průmyslu, vzdělávacích institucích a jiných podnicích a organizacích je zjevné. Téměř všechna povolání po celém světě závisí na softwarovém průmyslu pro vývoj produktu, produkci, marketingu, podpoře a službách. Snížení ceny vývoje a vylepšení kvality softwaru je důležité pro softwarový průmysl. Firmy hledají lepší způsoby jak provádět testování softwaru předtím než je nasadí.

Inovace na poli softwaru přinesly lepší techniky psaní testovacích skriptů, generování testovaných tříd a jiné nástroje testování. Účel těchto nástrojů je zrychlení životního cyklu vývoje softwaru, nalezení co nejvíce možných chyb před vypuštěním a zvýšení spolehlivosti aplikace. Zřejmý cíl nástrojů pro testování softwaru je generování skriptů, které simulují uživatelské používání aplikace již za vývoje. Obvykle testovací inženýři jsou trénováni vývojáři těchto nástrojů, aby psali testovací skripty ručně nebo aby použili zachytávací/přehrávací procedury k nahrávání testovacích skriptů. Psaní i nahrávání testovacích skriptů je pracné a náchylné k chybám. Potenciál dostupných nástrojů k redukci manuálního, opakovaného a pracného úsilí, potřebného pro test softwaru byl značně limitován. Naopak v dnešní době jsou automatické testovací nástroje jako TestComplete, který jsem používal, velice praktické a použitelné pro různé druhy testování.

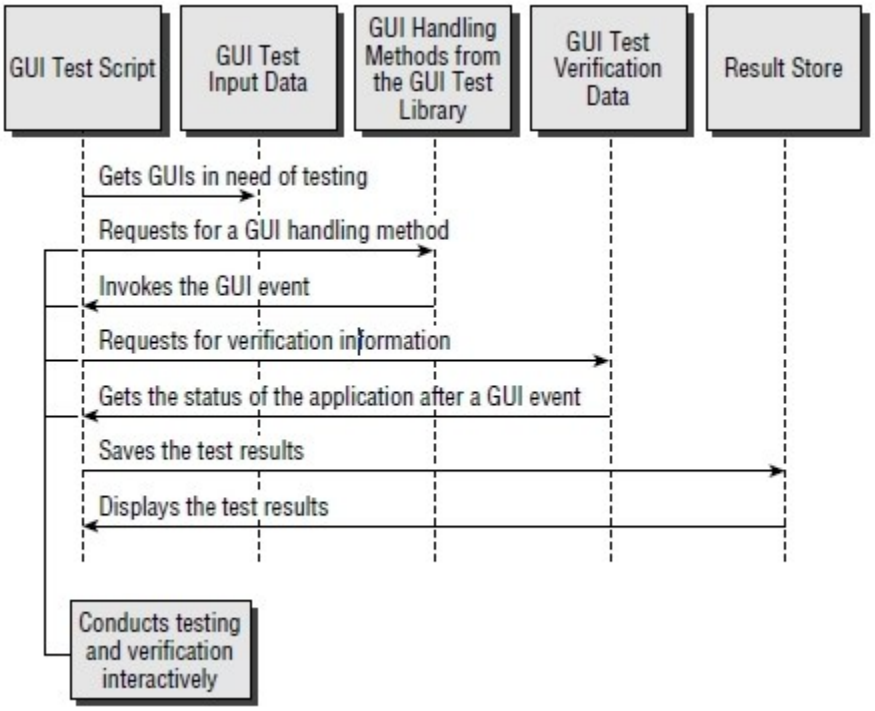
3.1 Co přináší automatické testování

„Dřívější softwarové aplikace byly ovládány příkazovou řádkou. Uživatelé si pamatovali a napsali příkaz a systém provedl nějaké akce. Efektivnější aplikace mohly zobrazovat možné příkazy na obrazovce a pak vyzvat uživatele k zadání jednoho z nich. V dnešní době je softwarový průmysl v éře oken. Dnes jsou všechny aplikace ovládány grafickým uživatelským rozhraním (GUI).“

(Li, a další, 2005)

Mezi GUI komponenty patří okna, ikony, menu a šipky. Uživatelé vyvolávají události tahem myší, klikáním na tlačítka a stlačováním různých kombinací kláves. Aplikace pak vykonávají očekávanou akci vyvolanou událostí myši a kláves.

Testování pomocí GUI je důležité, protože jsou prováděny jako zástup za koncového uživatele. Protože veškeré funkce aplikace jsou vyvolány skrze grafické rozhraní, GUI testování pak pokrývá celou aplikaci, viz Obrázek 1 : Dynamická daty řízená realizace a interakce mezi testovacím skriptem a datovým úložištěm (převzato z . Před obdobím GUI, testeři spoléhali na testovací skripty se souborem příkazů. Spouštění programů nebylo závislé na stavu obrazovky. Testování GUI komponent je ale rozdílné a obtížnější protože skript netestuje jen volání metod a událostí, ale také opravdu danou uživatelskou akci. Dále musí skript umět nahrávat odezvy softwaru a změny stavů. Porovnáním odezev a změn s očekávanými daty jsou skripty schopny ohlásit chyby.



.

Obrázek 1 : Dynamická daty řízená realizace a interakce mezi testovacím skriptem a datovým úložištěm (převzato z (Li, a další, 2005)

3.2 Očekávání automatického testování

Automatické testy značně pomáhají redukovat čas a náklady potřebné k testování softwaru během jeho vývoje. Dále automatizace zajišťuje pravidelnost a konzistenci, což pomáhá k dřívějšímu zjištění chyb v programu.

Současné programy nejsou schopny zautomatizovat testování skrze GUI před samotným vytvořením testovacích skriptů. Testeři často používají metodu nahrávání/přehrávání k nahrání těchto skriptů. Nahrávání je ve skutečnosti velice pracné. Tester musí postupně projít celým programem, nebo jeho částí, aby otestoval všechny tížené funkce. Pokud nastane nějaká chyba a nějaká část se nenahraje správně, musí opravit chybu zadání a pokračovat v nahrávání. V nejhorších případech znovu opakovat nahrání celého testu. Tester hledá nejlepší možnou cestu mezi testovaným a testovacím programem a touženém výsledném skriptu.

„Testeři očekávají od testovacího programu, že aktivně nalezne komponenty GUI testovaného programu, vygeneruje příslušná data a použije je k vytvoření a spuštění testovacího skriptu. I když prodejci často tvrdí že jejich program je schopen pomocí dat generovat automaticky skript, testeři často musí spouštět průvodce a přidat data ručně. Tento způsob skrz průvodce je mnohem pracnější a náchylný k chybám. První spuštění většinou není efektivní v nalézání chyb. Pro nalezení chyb by měl skript být schopen otestovat široké spektrum dat. Testeři očekávají automatizovanou metodu, která bude testovat mnoho různých testovacích dat.“ (Li, a další, 2005)

Komerčně dostupné testovací nástroje jsou velice závislé na platformě a většinou mají vlastní testovací prostředí. Nahraný test pak obvykle běží v prostředí, ve kterém byl nahrán. Nástroje jsou také doprovázeny skriptovacími jazyky. Pokud se chce kdokoli zabývat komplexním testováním softwaru, musí si zakoupit různé testovací nástroje od různých výrobců. Pro potřeby testování GUI je dostačující již dříve zmíněný program TestComplete.

Nástroj schopný efektivního nalezení chyb by měl mít tyto vlastnosti:

- Skriptovací jazyk bude stejný jako ten, který byl použit při vývoji aplikace.
- Snadná správa testování softwaru.
- Měl by být postaven na otevřené architektuře, což jej činí flexibilní a lehce modifikovatelný pro rozšířené testovací funkce.
- Regresivní testování by mělo být plně automatizováno.
- Výsledky testů by měly být udávány a ukládány v široce známých formátech.

4. Seznámení se softwarem

Seznamování probíhalo ve dvou krocích. První krok je teoretický. Představení softwaru vývojáři a pročtení dokumentace nápovědy v případě programu ArisCAT. Naučná videa o funkčnosti, pročtení manuálu a návody na internetu v případě programu TestComplete. Druhý krok je praktický. Projití potřebných modulů ve firemním programu a odzkoušení potřebných funkcí a schopností v programu testovacím.

4.1 ArisCAT

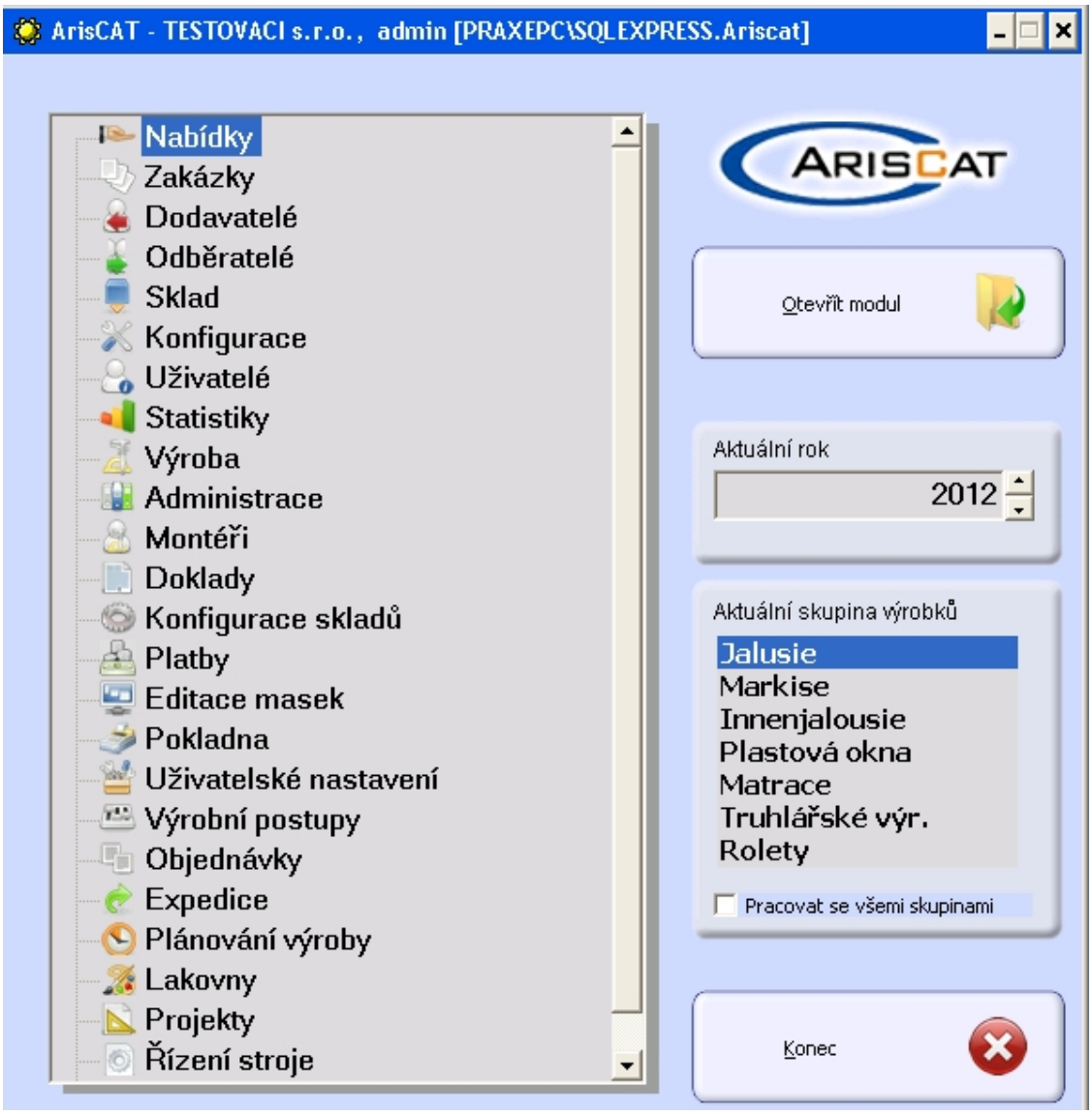
ArisCAT neboli Adaptabilní a Robustní Informační Systém firmy CATHEDRAL Software je kompletní informační systém pro obchodní a výrobní společnosti. Obsahuje rozšířenou možnost evidence zákazníků (CRM), jednoduchou a rychlou tvorbu nabídek, správu objednávek a řízení zakázkové výroby i výroby na sklad. Umožňuje vést neomezené množství skladů a organizovat materiál a zboží do skladových skupin. Manažerské výstupy a statistická data lze libovolně upravovat a exportovat do dokumentů Microsoft Office. Uživatel programu si může sám měnit tiskové sestavy a masky pro konfiguraci výrobků. (Cathedral Software, 2009)

Zákazníci mohou využít integrovaný internetový obchod, který si může upravit podle vlastních představ. Samotný program se skládá ze spousty modulů, viz Obrázek 2 : Úvodní okno programu ArisCAT.

Já jsem pracoval s těmito moduly:

- **Nabídky**, pro vytváření a editaci cenových nabídek, jejich tisk a možné přesunutí do modulu Zakázky
- **Zakázky**, pro vytváření, editaci a přehled vytvořených nebo již rozpracovaných zakázek a případně jejich následné přesunutí do modulu Výroba
- **Dodavatelé**, pro správu dodavatelů
- **Odběratelé**, pro správu odběratelů popřípadě nastavení individuálních parametrů jako např. slevy či příplatky pro jednotlivé výrobky, ceníky, měna
- **Výroba**, pro plánování výroby na základě objednávek a případnou expedici zboží
- **Expedice**, pro správu expedičních balíků

- **Konfigurace**, pro nastavení veškerých parametrů programu, definic výrobků, tiskových sestav a uživatelských akcí



Obrázek 2 : Úvodní okno programu ArisCAT

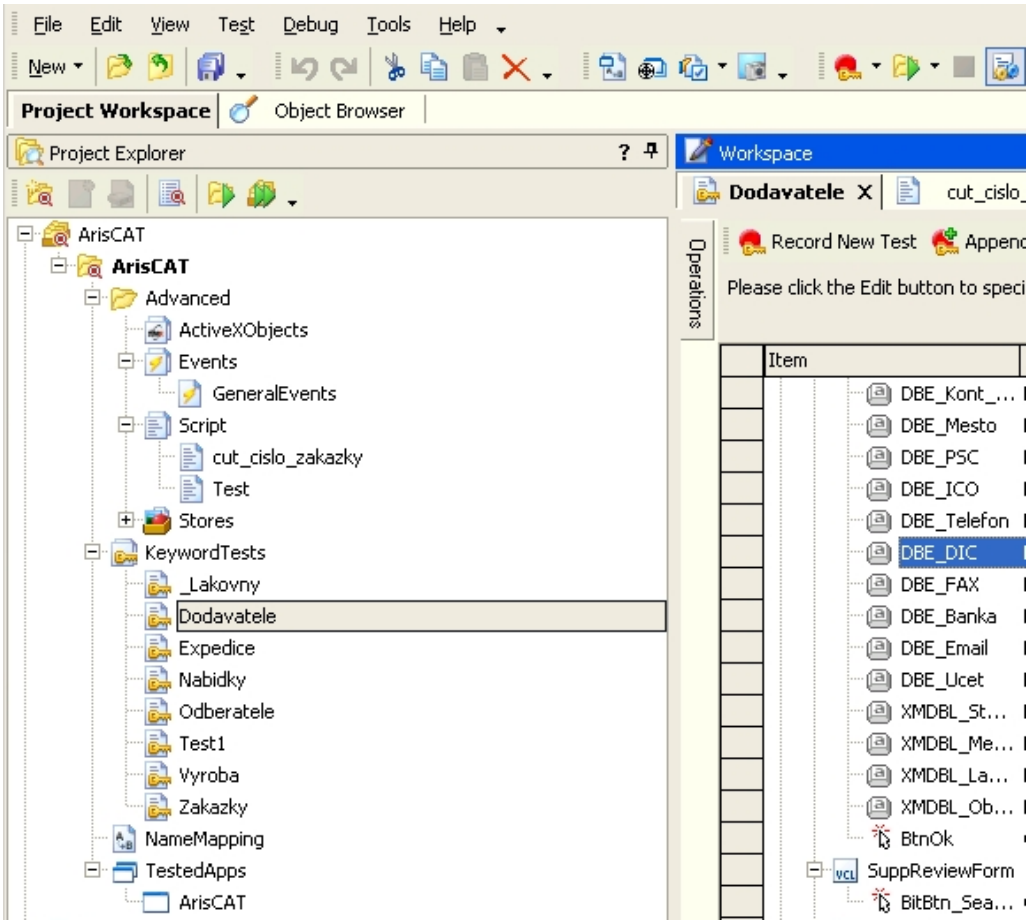
ArisCAT je velice rozsáhlý program pro nejen úzce specifikovanou sortu uživatelů. Uživatelé si mohou z dalších modulů vybrat k použití kromě již zmiňovaných třeba Doklady, Platby a Pokladna. Program se tímto stane jakousi pokladnou pro tohoto uživatele a ostatní moduly nemusí nutně používat. Díky rozsáhlému modulu Konfigurace si uživatel také může nastavit různá omezení, zjednodušení a spoustu parametrů. Mezi takové patří určitě skupiny výrobků, se kterými daná firma či zákazník pracuje a s tím spojená definice každého jednoho výrobku. Z dalších také nastavení měn, kurzů, slev z obratu a v neposlední řadě internetového portálu.

4.2 TestComplete

TestComplete od AutomatedQA's přináší spoustu možností testování softwaru. Pro mé potřeby ale postačí možnost automatického testování GUI pomocí testovacího skriptu a možnost vytvářet a psát si vlastní funkce ve skriptovacím jazyce. Mnou zvolený skriptovací jazyk je C# skript, který si program stejně upravuje do vlastního formátu.

TestComplete je plnohodnotné prostředí pro testování různých frameworků. V našem případě to bude Windows aplikace psaná v Delphi. Také možnost testování SQL serverů. Byl navrhován pro nezávislé vývojáře, aby omezil masivní spotřebu času a energie potřebnou k manuálnímu testování.

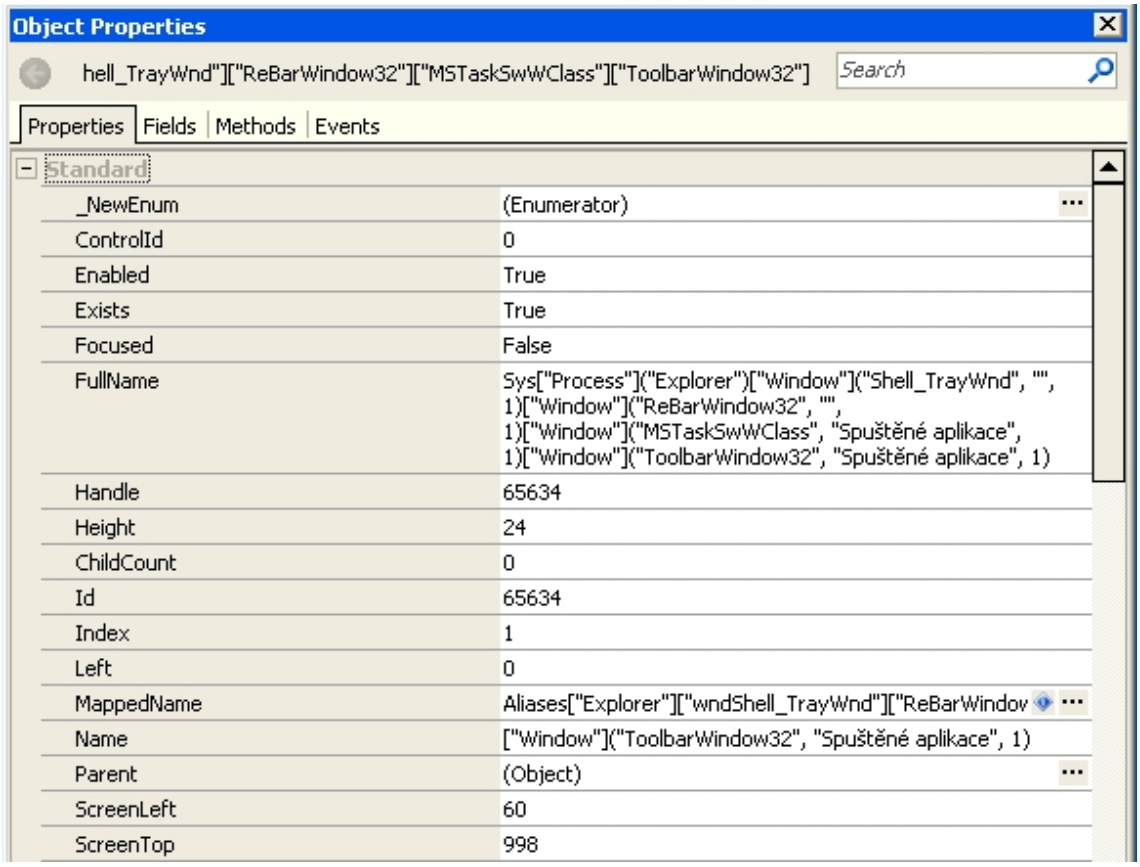
IDE programu TestComplete je velice obdobné s vývojovými nástroji. Na obrázku 2 je vidět jeho základní rozdělení.



Obrázek 3 : Hlavní okno programu TestComplete

Na Obrázek 3 : Hlavní okno programu TestComplete jsou patrné tyto důležité části:

- **project explorer** – zde jsou zobrazeny všechny součásti projektu jako třeba skripty, testy, testované aplikace, události a výsledky logu
- **object browser**– zobrazuje zásadní informace všech spuštěných programů a některé i měnit
- **code explorer** – je to jiný pohled na základ kódu, kde můžeme přehledně a snadno upravovat a mazat funkce zvláště ve velkých souborech
- **main workspace** – oblast zobrazující právě rozpracované skripty, testy či logy
- **finder tool** – nástroj k označení objektů přímo z obrazovky, viz Obrázek 4 : Okno Finder Tool



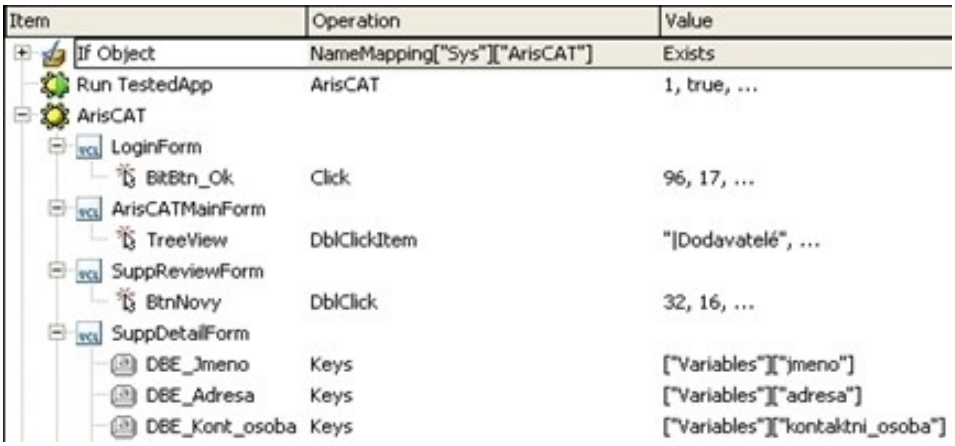
Obrázek 4 : Okno Finder Tool

5. Návrh a testování skriptovacích scénářů

Navrhování skriptovacích scénářů bylo v režii čisté teorie. Po praktickém seznámení s programem ArisCAT tedy nastal okamžik, kdy si člověk musí spojit veškeré znalosti, kterých nabyl během seznámení jak s testovaným tak s testovacím programem. Základním požadavkem na skriptovací scénáře bylo otestování základní funkčností zvolených modulů. Tedy v modulu nabídka šlo o vytvoření a uložení nabídky. Následovně kontrola konzistence dat a smazání testovací nabídky. Základní scénáře byly v textové formě a vypadaly takto:

otevřít modul nabídky
tlačítko nová - vytvořit novou nabídku
výběr organizace
tlačítko dále - vytvoření nabídky
výběr výrobku
tlačítko vybrat - zadání nabídky
...

Až všechny tyto scénáře byly schváleny zaměstnavatelem, přišla na řadu praktická implementace v prostředí TestComplete. Používal jsem již dříve zmiňovanou metodu zachytávací/přehrávací techniku pro nahrávání automatizovaného skriptu. V každém skriptu jsem musel vytvořit sadu proměnných, které je nutné fyzicky vyplnit nebo vybrat pomocí GUI. Nastavit vložení těchto proměnných do programu a naprogramovat jejich volání ve funkcích na kontrolu konzistence.



Item	Operation	Value
If Object	NameMapping["Sys"]["ArisCAT"]	Exists
Run TestedApp	ArisCAT	1, true, ...
ArisCAT		
LoginForm		
BitBtn_Ok	Click	96, 17, ...
ArisCATMainForm		
TreeView	DblClickItem	"|Dodavatelé", ...
SuppReviewForm		
BtnNovy	DblClick	32, 16, ...
SuppDetailForm		
DBE_Jmeno	Keys	["Variables"]["jmeno"]
DBE_Adresa	Keys	["Variables"]["adresa"]
DBE_Kont_osoba	Keys	["Variables"]["kontaktni_osoba"]

Obrázek 5 : Část automatizovaného skriptu

Důležitou součástí automatizovaného testu bylo vytvoření skriptovací funkce testující konzistenci dat mezi uložením a načtením viz Obrázek 6 : Část C# Skriptu. Před vymazáním tohoto objektu z databáze se proto zavolala tato funkce, která do logu zapsala úspěšnost kontroly dat.

Abych mohl psát tyto skripty, musel jsem se naučit syntaxi skriptovacího jazyka použitého v programu TestComplete. Ve vytváření těchto skriptů mi velice napomáhala funkce intellisense. Na webových stránkách je také velice důkladná dokumentace k používaným funkcím a metodám. Psaní těchto skriptů byla pracná neustále se opakující práce.

```
function test_Nabidky()
{
  Log["Message"]("Začátek testu Nabídky");
  if (Aliases["ArisCAT"]["DynForm"]["TopPanel"]["GrpBox_DaM"]["Pnl_HeO_Discounts"]["DBE_f
  if (Aliases["ArisCAT"]["DynForm"]["TopPanel"]["GRP_TGroupBox1"]["E_PocetSpojek_Plech"]["
  if (Aliases["ArisCAT"]["DynForm"]["XMZP_Grid"]["XMZoomPanelChild1"]["BottomPanel"]["E_P
  if (Aliases["ArisCAT"]["DynForm"]["XMZP_Grid"]["XMZoomPanelChild1"]["BottomPanel"]["GRP_
  if (Aliases["ArisCAT"]["DynForm"]["XMZP_Grid"]["XMZoomPanelChild1"]["BottomPanel"]["GRP_
  if (Aliases["ArisCAT"]["DynForm"]["XMZP_Grid"]["XMZoomPanelChild1"]["BottomPanel"]["GRP_
  if (Aliases["ArisCAT"]["DynForm"]["XMZP_Grid"]["XMZoomPanelChild1"]["BottomPanel"]["E_B
  if (Aliases["ArisCAT"]["DynForm"]["XMZP_Grid"]["XMZoomPanelChild1"]["BottomPanel"]["E_B
  if (Aliases["ArisCAT"]["DynForm"]["XMZP_Grid"]["XMZoomPanelChild1"]["BottomPanel"]["E_D
  Log["Message"]("Konec testu Nabídky");
}
```

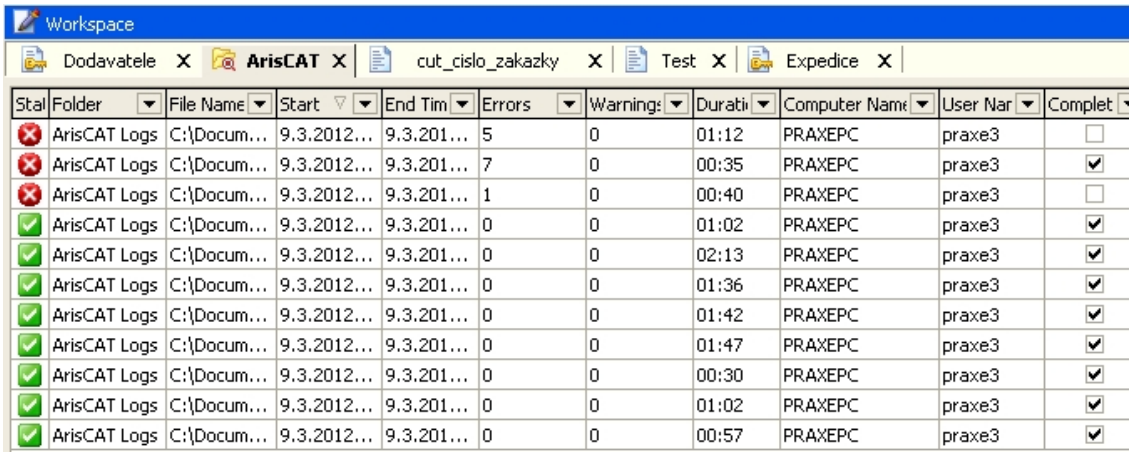
Obrázek 6 : Část C# Skriptu, který testuje správnost uložených dat v programu se zadanými

Při vytváření automatizovaných skriptů pro další moduly docházelo k duplicitám velkých částí skriptu. TestComplete umožňuje v jednom skriptu si zavolat druhý již dříve vytvořený. Což bylo na první pohled velice praktické. Ale po krátkém zvážení jsem si uvědomil, že tyto části raději zkopíruji, jelikož v dříve napsaných skriptech docházelo k mazání původních dat. Což by mi znemožnilo navázání v novém skriptu.

Při vytváření skriptů také vyšly najevo nedostatky programu TestComplete a nebo naopak vysoké složitosti programu ArisCAT. Při dynamickém generování některých datagridů v programu ArisCAT nebyl schopný TestComplete zjistit pozici a nalézt daný prvek. Test se tím pokaždé zastavil a nechtěl pokračovat. Po relativně krátké konzultaci s programátory jsme problém vyřešili. Ve skriptu se musel volat prvek, ve kterém byl datagrid zapouzdřen.

5.1 Testování skriptovacích scénářů

Testování automatizovaných skriptů je značně zdlouhavé. Člověk jen kontroluje průběh testu a čeká na výsledek. Výhodou je log chyb, ve kterém se vypíše jakákoliv i sebemenší chyba. Není tedy potřeba u testu celou dobu sedět. Pokud vše proběhlo dle požadavků a bez problémů ve výpisu logu byl ihned vidět výsledek. Viz Obrázek 7 : Výpis logů.



Stal	Folder	File Name	Start	End Tim	Errors	Warnings	Durati	Computer Name	User Nar	Complet
✗	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	5	0	01:12	PRAXEPC	praxe3	☐
✗	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	7	0	00:35	PRAXEPC	praxe3	☑
✗	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	1	0	00:40	PRAXEPC	praxe3	☐
✓	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	0	0	01:02	PRAXEPC	praxe3	☑
✓	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	0	0	02:13	PRAXEPC	praxe3	☑
✓	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	0	0	01:36	PRAXEPC	praxe3	☑
✓	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	0	0	01:42	PRAXEPC	praxe3	☑
✓	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	0	0	01:47	PRAXEPC	praxe3	☑
✓	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	0	0	00:30	PRAXEPC	praxe3	☑
✓	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	0	0	01:02	PRAXEPC	praxe3	☑
✓	ArisCAT Logs	C:\Docum...	9.3.2012...	9.3.201...	0	0	00:57	PRAXEPC	praxe3	☑

Obrázek 7 : Výpis logů ukončených testů z programu TestComplete

Pokud při testu nastane nějaká chyba, zastaví se a chyba se vypíše do logu. Podle možností nastavení v programu TestComplete dojde k přerušení testu ihned nebo až po několika opakovaných pokusech. Podle výpisu logu člověk pak musí projít skript, nalézt chybný kód a opravit jej. Nakonec skript musí opět proběhnout, aby se zjistilo, jestli je chyba správně opravena.

Testování skriptů je v první řadě náročné na software. Záleží tedy na tom jak je dobře napsaný kód programu a co je nutné při každé interakci načíst použít a poslat na server. Pokud je toho příliš tak má program pomalé odezvy a testování je pak zbytečně zdlouhavé. V druhé řadě také hodně záleží na hardwaru. Čím rychlejší je počítačová sestava, na které je testování spuštěno, tím bude rychlejší. Ale ani nejrychlejší počítač světa nezrychlí datové připojení na server a špatně optimalizovaný kód.

6. Návrh dalšího rozvoje problematiky

Na několika posledních schůzkách jsme s firmou řešili nasazení testovacích skriptů do jejich cyklu vývoje softwaru. Řešilo se nasazení a provázání se systémem pro správu verzí svn, se kterým firma již po nějakou dobu pracuje. I když skripty programu TestComplete jsou založené na xml, dají porovnávat svn systémem. To ale bude dále už v režii firemních programátorů, aby provázali funkce těchto programů.

Další z návrhů je rozšíření původních testovacích skriptů. Původní mnou psané skripty testovaly pouze základní funkčnosti modulů programu. Pokud by se programátoři rozhodli pro testování různých vstupů, špatných vstupů či snad testovat víceuživatelský režim programu, museli by skripty rozšířit nebo vytvořit jiné složitější na základě již vytvořených.

Dále jsme s programátory objevili možnost skriptovacího testování SQL serverů. Ve skriptovací části programu TestComplete se dají vytvářet i volat složité SQL příkazy a také zachytávat odpovědi serveru. Jelikož program také pracuje se serverovým přenosem dat, je tahle schopnost testovacího nástroje také použitelná do budoucího vývoje.

7. Závěr

Cílem individuální praxe tedy byla automatizace testovacího procesu během vývoje firemního softwaru ArisCAT. Firma Cathedral Software měla k těmto účelům již zakoupený software s názvem TestComplete. Vše bylo tedy otázkou člověka, který se s daným programem naučí. Vytvoří automatizované testy a otestuje tím program ArisCAT. Dále pak prezentace své práce dalším programátorům pro další rozvoj.

Volba testovacího programu firmou je plně dostačující. Program TestComplete totiž zvládá nejen původní nároky firmy, ale i mnohé navíc. Pokud by se firma zajímala o detailnější testování vyvíjeného programu, mohou se jednoduše doučit další možnosti testování pomocí nástroje TestComplete.

I když firma sídlí značně daleko a musel jsem tedy pracně dojíždět, nikdy nebyl problém se spojením ani se zadáním práce. Vše bylo vyřešeno vzdálenou správou firemního serverového počítače, kde byly nainstalované oba programy. Dále také emailovým a telefonním kontaktem. Samozřejmě také v předem naplánovaných schůzkách, na kterých jsme sledovali postupný vývoj práce a navrhovali další úkoly.

Testovací skripty jsem vytvářel a testoval na serverovém počítači. Testování bylo spouštěno postupně pro každý modul zvlášť. Pokud by firma toužila automatizovat celý proces, dají se všechny skripty sloučit do jednoho. Odladění veškerých skriptů mě stálo spoustu hodin práce a sedění u počítače. Kontrola logů je rychlá, ale průběh každého testu byl časově náročný z pohledu práce na počítači. Naopak z pohledu testování softwaru je tento čas minimální.

Úplným zakončením mé praxe bylo předání vytvořených odzkoušených testovacích skriptů. Dále prezentace celé mé práce a vysvětlení základní funkce programu TestComplete. Dále jsme prodiskutovali další možnosti testování jejich programu ArisCAT.

8. Citovaná literatura

Cathedral Software. 2009. *ArisCAT help.* Prostějov : autor neznámý, 2009.

Li, Kanglin a Mengqi, Wu. 2005. *Effective GUI Test Automation: Developing an Automated GUI Testing Tool.* United States of America : Sybex, 2005. str. 445.

Tadros, Lino a Trefethen, Steve. 2009. *TestComplete 7 Made Easy.* místo neznámé : AutomatedQA Corp., 2009. str. 302.